C. AMENDMENTS TO THE CLAIMS

In order to better assist the Examiner with the prosecution of the case, the current pending claims have been included in their entirety for which reconsideration is requested.

1. (AMENDED) A computer implemented method in a virtual machine layer of ~~operating~~ a ~~computing~~ computer system supporting multiple processes, said method including the steps of:

~~providing~~ utilizing a set of monitors for controlling access to resources of the computer system, whereby a given process has to enter a given monitor in order to access a corresponding given resource, and has to join an entry queue for the given monitor if the given resource is currently owned by another process for which the given process has entered;

responsive to a predetermined condition of a requesting process requesting to enter a requested monitor, from the set of monitors, currently owned by a different process, examining queued processes queued on at least one of the set of monitors to determine whether there is a deadlock situation by detecting a cyclic dependency at a time a given dependency is created; and

if ~~such a~~ the deadlock situation is found, returning information about ~~the~~ an identity of ~~the~~ i) identified processes from the examined queued processes, and ii) identified monitors from the set of monitors, involved in the deadlock situation with an exception returned to an application while the application is running on the virtual

machine layer of the computer system thereby enabling the application to resolve the deadlock situation by having at least one of the identified processes release at least one of the identified monitors that the at least one identified process owns but is not currently using for continued processing by the application without the deadlock situation.

2. (CANCELED)

3. (AMENDED) The computer implemented method of claim ~~2~~ 1, wherein the examined queued processes involved in said examining step are determined iteratively based on ~~those~~ at least one of the multiple processes queued on the requested monitor, and ~~those~~ at least one of the multiple processes queued on other monitors owned by examined queued processes already determined to be involved in said examining step.

4. (AMENDED) The computer implemented method of claim ~~2~~ 1, wherein the system maintains a global table of contended monitors, a given one of the contended monitors ~~monitor~~ being contended if there is a given queued process in its entry queue, the table further identifying those processes which own or are queued on contended monitors.

5. (CANCELED)

6. (CANCELED)

7. (CANCELED)

8. (CANCELED)

9. (AMENDED) The computer implemented method of claim 8 1, wherein ~~said predetermined condition comprises a process requesting to enter a monitor that is currently owned by another process, and~~ said exception is returned to the requesting process.

10. (AMENDED) The computer implemented method of claim 8 1, wherein the exception is returned to all identified processes involved in the deadlock situation.

11. (AMENDED) The computer implemented method of claim 1, wherein the examination of queued processes queued on the at least one of the set of monitors to determine whether there is [a] the deadlock situation includes those processes having a conditional wait on [a] at least one of the set of monitors ~~monitor~~.

12. (AMENDED) The computer implemented method of claim 1, wherein the examination of queued processes queued on at least one of the set of monitors includes those monitors on ~~one or more~~ at least one remote machine ~~remote machines~~.

13. (AMENDED) A ~~computing~~ computer system having a virtual machine layer between applications executing on the computer system and an operating system of the computer

system, the computer system having means for supporting multiple processes, ~~and~~ ~~including~~ the computer system comprising:

a set of monitors for controlling access to a plurality of resources of the computer system, whereby a given process has to enter a given monitor in order to access a corresponding given resource~~, and~~;

~~has to join~~ an entry queue for ~~the~~ each given monitor if the corresponding given resource is currently owned by another process for which the given process has entered;

means, within the virtual machine layer of the computer system, responsive to a predetermined condition of a requesting process requesting to enter a requested monitor, from the set of monitors, currently owned by a different process for examining queued processes queued on at least one of the set of monitors to determine whether there is a deadlock situation by detecting a cyclic dependency at a time a given dependency is created; and

means, responsive to ~~such~~ the deadlock situation being found, for returning information about ~~the~~ an identity of ~~the~~ i) identified processes from the examined queued processes, and ii) identified monitors from the set of monitors, involved in the deadlock situation with an exception returned to a given one of the applications while the application is running on the virtual machine layer of the computer system thereby enabling the application to resolve the deadlock situation by having at least one of the identified processes release at least one of the identified monitors that the at least one

identified process owns but is not currently using for continued processing by the application without the deadlock situation.

14. (CANCELED

15. (AMENDED) The system of claim ~~14~~ 13, wherein the examined queued processes to be examined are determined iteratively based on ~~those~~ at least one of the multiple processes queued on the requested monitor, and ~~those~~ at least one of the multiple processes queued on other monitors owned by examined queued processes already determined to be examined.

16. (AMENDED) The system of claim ~~14~~ 13, wherein the system maintains a global table of contended monitors, a given one of the contended monitors ~~monitor~~ being contended if there is a given queued process in its entry queue, the table further identifying those processes which own or are queued on contended monitors.

17. (CANCELED)

18. (CANCELED)

19. (CANCELED)

20. (CANCELED)

21. (AMENDED) The system of claim ~~20~~ 13, wherein ~~said predetermined condition comprises a process requesting to enter a monitor that is currently owned by another process, and~~ said exception is returned to the requesting process.

22. (AMENDED) The system of claim ~~20~~ 13, wherein the exception is returned to all identified processes involved in the deadlock situation.

23. (AMENDED) The system of claim 13, wherein the examination of queued processes queued on the at least one of the set of monitors to determine whether there is [a] the deadlock situation includes those processes having a conditional wait on [a] at least one of the set of monitors ~~monitor~~.

24. (AMENDED) The system of claim 13, wherein the examination of queued processes queued on at least one of the set of monitors includes those monitors on ~~one or more remote machines~~ at least one remote machine.

25. (AMENDED) A computer program product comprising program instructions encoded in machine readable form on a medium, said instructions when loaded into a computer system, having means to support multiple processes, causing a virtual machine layer of the computer system to perform the steps of:

~~providing~~ utilizing a set of monitors for controlling access to resources of the computer system, whereby a given process has to enter a given monitor in order to access a corresponding given resource, and has to join an entry queue for the given monitor if the given resource is currently owned by another process for which the given process has entered;

responsive to a predetermined condition of a requesting process requesting to enter a requested monitor, from the set of monitors, currently owned by a different process, examining queued processes queued on at least one of the set of monitors to determine whether there is a deadlock situation by detecting a cyclic dependency at a time a given dependency is created; and

if ~~such a~~ the deadlock situation is found, returning information about ~~the~~ an identity of ~~the~~ i) identified processes from the examined queued processes, and ii) identified monitors from the set of monitors, involved in the deadlock situation with an exception returned to an application while the application is running on the virtual machine layer of the computer system thereby enabling the application to resolve the deadlock situation by having at least one of the identified processes release at least one of the identified monitors that the at least one identified process owns but is not currently using for continued processing by the application without the deadlock situation.

26. (CANCELED)

27. (AMENDED) The computer program product of claim ~~26~~ 25, wherein the examined queued processes involved in said examining step are determined iteratively based on ~~those~~ at least one of the multiple processes queued on the requested monitor, and ~~those~~ at least one of the multiple processes queued on other monitors owned by examined queued processes already determined to be involved in said examining step.

28. (AMENDED) The computer program product of claim ~~26~~ 25, wherein the system maintains a global table of contended monitors, a given one of the contended monitors ~~monitor~~ being contended if there is a given queued process in its entry queue, the table further identifying those processes which own or are queued on contended monitors.

29. (CANCELED)

30. (CANCELED)

31. (CANCELED)

32. (CANCELED)

33. (AMENDED) The computer program product of claim ~~32~~ 25, wherein ~~said predetermined condition comprises a process requesting to enter a monitor that is currently owned by another process, and~~ said exception is returned to the requesting process.

34. (AMENDED) The computer program product of claim ~~32~~ 25, wherein the exception is returned to all identified processes involved in the deadlock situation.

35. (AMENDED) The computer program product of claim 25, wherein the examination of queued processes queued on the at least one of the set of monitors to determine whether there is [a] the deadlock situation includes those processes having a conditional wait on [a] at least one of the set of monitors ~~monitor~~.

36. (AMENDED) The computer program product of claim 25, wherein the examination of queued processes queued on at least one of the set of monitors includes those monitors on ~~one or more~~ at least one remote machine ~~remote machines~~.